Halmstad Colloquium talk 2013-02-24:

Ty type or not to type…….

by **professor Robert Cartwright, Rice University, Houston, USA**

Abstract:
When I started programming in 1967, program types referred to machine representation formats. The high-level languages of that time (e.g., Fortran) relied on type declarations to determine the meaning of program operations like '+' and '*'. In contrast, contemporary type systems for programming languages are intricate mathematical constructions that can prove theorems about program behavior. They also constitute an important issue in "language wars" about which language is best for a particular application. In this talk, I will review the evolution of type systems since the 1967 and present a dispassionate account (identifying my opinions in some cases) of the advantages and disadvantages of various approaches to program typing. In the process, I will identify a few of my contributions to the narrative. I try to focus on the roles that type systems play in software engineering and their interaction with program design.

Biography:
Robert “Corky” Cartwright has been a professor of Computer Science at Rice University since 1980. He earned a bachelor’s degree magna cum laude in Applied Mathematics from Harvard College in 1971 and a doctoral degree in Computer Science from Stanford University in 1977. Prior to joining the Rice faculty, he worked as an assistant professor at Cornell University. Throughout his career, his principal professional goal has been elevating programming from a black art to a systematic discipline. Professor Cartwright's principal research interests are programming language design and implementation, program specification, program testing and analysis, and software engineering. He is currently engaged in five major research projects: (i) Soft Typing: developing program analysis tools for Java that use precise type inference to help programmers debug and optimize programs; (ii) Dr Java and DrScala : constructing production quality, open source, pedagogic programming environments for Java and Scala that foster test-driven software development; (iii) Testing Frameworks for Concurrent Programs: creating tools to support the systematic testing of Java programs with multiple threads of control; (iv) Simulation Frameworks for Hybrid Systems: developing new languages, environments, and simulation techniques for designing and prototyping systems where computer software interacts with physical components (e.g., robots); and (v) Denotational Models of Object-Oriented Languages: developing accurate semantic models of object-oriented languages like Java, C#, and Scala.

From 1991-1996 Professor Cartwright served as a member of the ACM Turing Award Committee, which selects the annual recipient of the most prestigious international prize for computer science research. In 1998, he was elected as a Fellow of the Association for Computing Machinery (ACM), the leading professional organization for computer scientists. From 1991-1996 he served as a member of the ACM Turing Award Committee, which selects the annual recipient of the most prestigious international prize for computer science research. He served as a member of the Board of Directors for the Computing Research Association from 1994-2000 and helped organize the Coalition to Diversity Computing. In 1998, he was elected as a Fellow of the Association for Computing Machinery (ACM), the leading professional organization for computer scientists. He has also served as a member of ACM Education Board from 1997-2006 and a member of Sun Microsystems Developer Advisory Council from 2002-2009. He currently serves a member of the Computer Science Advisory Council at Prairie View A&M University.